## APPENDIX 1

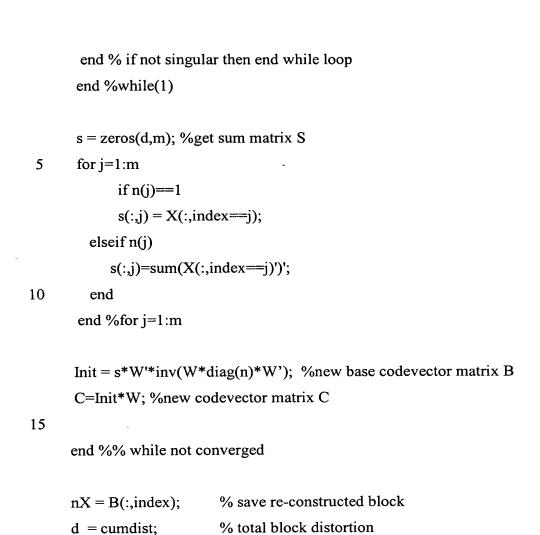
```
%%%
     function B = Initcluster(X,m)
5
     % Get an initial base codebook B at random from input data
     % INPUTS
     % X = input data: each column is a RGB 'vector'
     % m = number of base codevectors
10
     % OUTPUTS
     \% B = base codevector matix
     %need to duplicate some columns of B if X is small
     [n,N] = size(X);
15
      if(N > m)
       replace = 0;
      else
       replace = 1;
20
      end
      %track what inputs put in B, so no duplication for X large
      chosen = zeros(1,N);
      B = zeros(n,m);
25
      for i=1:m
       draw = floor(N*rand + 1);
       if(~replace)
        while(chosen(draw))
          draw = floor(N*rand + 1);
30
         end
```

```
end
      B(:,i) = X(:,draw);
      chosen(draw) = 1;
     end
5
     %%%
     function [B,Bq,d,iters,rd,nX] = lgbo(X,m,base)
10
     % INPUTS
      % X = input data: each column is a RGB 'vector'
      % m = number of codevectors (columns in C)
      % base = number of base codevectors (columns of B)
      % OUTPUTS
15
      \% B = base codevector matrix
      % Bq = quantized base codevector matrix
      % d = distortion of X when replaced with chosen codevectors
      % iters = # of iterations to reach convergence
20
      [d,N] = size(X); % d = dimension, N = blocksize
      %random initialization
                                 % choose random initial set
      Init = initcluster(X,base);
25
                     %1D (linear interpolation)
      if base==2
             W = [1:(-1/(m-1)):0; 0:(1/(m-1)):1]; %weight matrix
      elseif base=3 %2D
              W=[1,0,0;0,1,0;.25,.25,.5;-.25,-.25,1.5;...]
               1,-1,1;-1,1,1;0,-1,2;-1,0,2;]';
30
```

```
elseif base==4 %3D
             W=[eye(4),.25*[2,2,1,-1;2,-1,2,1;1,2,-1,2;-1,1,2,2]];
        %W=[eye(4),.125*[3,2,2,1;2,3,2,1;3,1,2,2;1,2,3,2]];
     end
5
     %variable initialization
     stoppingeps = 1.e-5;
     vi = ones(1,m);
     index=zeros(1,N);mind=index;
10
     cumdist = Inf; lastdist = 0;
      C=Init*W; %interpolate codevectors
      # iterate until convergence
      while(abs(cumdist - lastdist) > stoppingeps)
15
        lastdist = cumdist;
        cumdist = 0;
        iters=iters+1;
        while(1) %1 iteration that repeats if B goes singular
20
         % step (A)
         % form Voronoi regions: for each input,
             % determine which centroid it is closest to
             for i=1:N
25
          V=X(:,i(vi))-C;
          nm=sum(V.*V); %Euclidean distance squared (MSE)
          %input i's closest codevector
          [mind(i),index(i)]=min(nm);
         end
30
         cumdist = sum(mind);
```

```
%find diagonal matrix N
        for j=1:m
             n(j) = sum(index==j);
5
        end
        % check if B is singular
        % and force it to be non-singular
        num=sum(n\sim=0);
        if num<br/>
<br/>
base %fewer than base non-zero!
10
         nz = find(n);
              if (base-num)==1 %base==2
          [jy,ji]=max(mind);
          Init(:,1:(base-1))=B(:,nz);
15
          Init(:,base)=X(:,ji);
          else
          [jy,ji]=sort(mind);
          Init(:,1:num)=B(:,nz);
          jl=N;
20
           for jk=(num+1):base
            Init(:,jk)=X(:,ji(jl));
            while(sum(abs(X(:,ji(jl))-X(:,ji(jl-1))))==0)
              jl=jl-1;
             end
25
            jl=jl-1;
           end % for jk
          end % else
              C=Init*W; %start all over if hit a singular matrix
        else
30
         break;
```

20



end % function